Giga-particle trajectory model

The Giga-particle trajectory model implements a particle trajectory program. Basically it integrates the trajectory equation

d **r** /dt = **u** (**r** ,t)

The original version is implemented in a proprietary interpreted language called IDL. The new version is written in C++ to allow the program to be implemented as an extensible application framework and to make use of the faster execution of a compiled language. The program is also parallelized using the Message Passing Interface (MPI) library to allow the simulation of 1 billion trajectories on a suitable NASA computing platform.

This user guide describes how to download, compile and run the C++ version of the model.

The model code is available from the CVS repository at http://sourcemotel.gsfc.nasa.gov.

Before downloading the program, it is useful to start by setting the following environment variables in order to specify the remote login program and to point to the location of the root directory of the source code.

For example, if you are using csh:

```
setenv CVS_RSH ssh
setenv CVSROOT  <login
name>@sourcemotel.gsfc.nasa.gov:/cvsroot/gtraj
```

The entire program along with unit tests and test data can be downloaded as follows:

```
cvs co gtraj
```

The root directory is gtraj. Under gtraj there are several subdirectories, including:

- src directory: This directory contains C++ source files, header files and build scripts for the Gtraj program.
- test directory: This directory contains C++ source files, header files and build scripts for CppUnit testing of class methods.
- doc directory: This directory contains documentation files.
- etc directory: This directory contains scripts and other configuration for running the model and test cases.

==

How to run tests:

==

On DISCOVER:

Before attempting to compile you must setup your computational environment. For

example if you use the csh shell, do the following:

```
   source /usr/share/modules/init/bash
   module purge
   module load comp/intel-9.1.052 lib/mkl-9.1.023 mpi/impi-3.1.038
   setenv LD_LIBRARY_PATH
$LD_LIBRARY_PATH:/usr/local/other/cppunit/1.12.1_gcc4.1.1/lib
   setenv CPPUNIT /usr/local/other/cppunit/1.12.1_gcc4.1.1
```

Then:

1) From the top level directory (gtraj) type:

```
   make test MODE=SERIAL
```

This will build and run the tests on 1-CPU. For other build options

type "make help". If no options are specified the code will build in MPI mode.

Alternatively

```
   make install MODE=SERIAL
```

will build/install the model and build the tests.

2) When done

```
   cd Linux/bin
   gtraj-test.j << edit as needed to locate gtraj data and
configuration file
```

To run in paralle submit job to queue as follows:

```
   qsub gtraj-test.j << edit RUNCMD setting
```

==

How to run the model:

==

The model can be run as-is with a default configuration. However some data sets are needed (and include the default configuration). These can be obtained on DISCOVER from /discover/nobackup/ccruz/GTRAJ. Grab:

```
gtraj_DATA.tgz
gtraj_REAN.tgz

and

tar xvfz gtraj_DATA.tgz
tarxvfz gtraj_REAN.tgz
```

to your work space. The first file, gtraj_DATA.tgz, conatins parcel locations files that can be used to read in initial parcel positions at the beginning of a simulation. The second file, gtraj_REAN.tgz, contains grid data variable files hold metrological data sets for particular variables at regular timesteps.

1) After installation and data-setup is complete, simply

```
  cd Linux/bin
  gtraj.j  << edit as needed to locate gtraj data and
configuration file
```

To run in paralle submit job to queue as follows:

```
qsub gtraj.j << edit RUNCMD setting
```

==

On other Unix-like platforms:

==

1) Install CPPUNIT software.

2) Add the current directory and the cppunit software library location to

the load library path. For example:

if

```
LD_LIBRARY_PATH="/Users/"$USER"/libs/cppunit-1.12.1/lib"
CPPUNIT_PATH="/Users/"$USER"/libs/cppunit-1.12.1"
```

then

```
setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH
setenv CPPUNIT $CPPUNIT_PATH
```

(on Darwin, setenv DYLD_LIBRARY_PATH $DYLD_LIBRARY_PATH)

3) Compile and run the tests/model as above.

==

Configuring the model

==

A default configuration file, “gtraj.conf”, is located in the data directory and will be used if none is specified. The following parameters can be modified in gtraj.conf:

Parameter Name	**Description and values**
DATE_STEP_SIZE	The time interval in days metrological data is read in. For example the value could be 0.4 days (or 6 hours). This means the metrological data is available at every 6-hour intervals.
DIRECTION	Direction of the simulation in time: • 1 = forward in time • -1 = backward in time
DURATION	Number of iterations to run the simulation.
TRAJECTORY_TYPE	Type of physics process to use when running the simulation. • 0 = isentropic • 1 = kinematic

	• 2 = neither
DT	Time step in number of days. The clock is incremented or decremented by “dt” every iteration. The clock is incremented for forward simulations in time and decremented for backward simulations in time. A typical value is 0.05.
START_DATE	Date and time of the start of the simulation in the form: yyyy-mm-ddThh. For example: 2006-04-30T18
START_TIME	An arbitrary real value that corresponds to the “START_DATE”. Time is measured relative to this value. For a forward simulation it is set to 0. For a backward simulation it is set to some arbitrary positive value.
CURRENT_TIME	A real value that denotes the starting time of the simulation. For a forward simulation it is set to a value in the range START_TIME, START_TIME + DATE_STEP_SIZE. For a backward simulation it is set to a value in the range START_TIME – DATE_STEP_SIZE, START_TIME.
LOCATION	Directory path where data files for experiment are stored.
LABEL	This field is currently not used.
RANDOM_SEED	A value used to seed the random number generator when used.
NPARCELS	Total number of parcels to simulate.
PRINT_PARCELS	Adds a header line for printing parcels.
INIT_METHOD	Method used to initialize the parcel locations. • 0 = randomly selected locations • 1 = locations from file
GRID_TYPE	Type of grids available: • ReversedGrid = a grid whose abscissa values could be in reverse order. • ZGrid = a grid whose abscissa values in the z-direction are irregular, unordered and changes with time.
FIELD_TYPE	Type of interpolation fields: InterpolatedVectorField
CONDITIONER_TYPE	Type of conditioners available: • Corrector = works with parcel data to set the out-of-bound values of longitude, latitude and height to the appropriate range after each step of the 4-th order Runge Kutta simulation.

INTEGRATOR_TYPE	Type of integration algorithms available: RungeKutta
GRID_CONDITIONER	Type of grid conditioners available: • Filter = corrects for bad or missing metrological data. • Periodic = conditioner implements periodic boundary conditions along the longitude direction.
DATA_SOURCE_TYPE	Type of data sources available: DataFile
INTERPOLATOR_TYPE	Type of interpolator for interpolating metrological data to parcel positions. This could be a linear interpolator, quadratic interpolator or cubic spline interpolator. Currently only the linear interpolator is supported but a template has been implemented to show how to implement quadratic or cubic spline type interpolators.
TIME_INTERPOLATOR_TYPE	Type of interpolator to be used for time interpolation of metrological data. Usually this will be a linear interpolator.
OUTPUT_GRID_DATA	These groups of parameters, identified with a number at the end, define the meteorological properties for generating the output. Each grid data unit can define several types of parameters within the block: • TYPE = the name of the class to create • PROPERTY = a specific meteorological property (e.g. pressure, temperature) • DEPENDENCY_PROPERTY = a meteorological property required to calculate another property Note that “OUTPUT_GRID_DATA.0” will be considered the “source” for dependent meteorological properties.
OUTPUT_INTERPOLATOR	This group of parameters defines how the interpolator is configured for output data, The output interpolator identifies the metrological properties and grid dependencies through the following parameters: • GRID_DATA_INDEX – specifies the index of the OUTPUT_GRID_DATA being used; all parameters below the index are associated with this particular grid data • PROPERTY – meteorological property to output
DEPENDENCY_GRID_TYPE	identifies the type of grid being configured
OBSERVING_PROCESSES	A list of observing processes that can be added to the simulation for additional calculations, storage, or visualization. These include: InterpolatedOutputObserver, DayLightObserver, StatsObserver. Refer to the Observer Class description for details on these.

==

Additional documentation

==

Additional documentation is generated opon installation (make install). This is found under gtraj/doc/html and can be viewed using any browser by opening the index.html file (note that there will be several files under html).